**API – updated after part 2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EXPLANATION** | **RETURNS** | **PARAMETERS** | **HTTP METHOD** | **METHOD NAME** | **ID** |
| Login cannot pass in GET. | {  "canLogin": bool,  "lastLoginDate":Date  }  (True= login successfully.  False =login fail) | {  "username": string,  "password"string:  } | POST | loginUser | 1 |
| Disks are related to the seller not to the client.  Using GET because there is no information to deliver to the server or any wish to change any information. | Returns array of 5 most popular Disks of the week.  HotDisk Json:  {  "pID":int ,  " artist ":string ,  "name": string ,  "cost":real ,  "purchaceAmount": int,  "addDate": date ,  "publishDate":date,  "totalWeekAmount"int:  } |  | GET | fiveHotDisks | 2 |
| Register cannot pass in GET.  There is information to deliver to the server and also update the db. | Bool  (True= registered successfully.  False =registered fail, username already exist) | {  "username": string,  "password": string,  "firstName":string,  "lastName":string",  "email": string,  "restoreAnswer":string,  "country":string,  "address": string,  "city": string,  "favoriteCatagoriesList": array  “creditCardNumber”: string  } | POST | registerNewUser | 3 |
| restorePassword cannot pass in GET, because we want to send the restore data from client to server. | string  (if user and answer are correct: return password,  If user exist but answer wrong: return “wrong answer”,  If user does not exist: return “wrong user”) | {  "username": string,  "restoreAnswer":string  } | POST | restorePassword | 4 |
| Disks are related to the seller not to the client.  Using GET because there is no information to deliver to the server or any wish to change any information. | Returns array of newest Disks of the last month.  Disk Json:  {  "pID":int ,  " artist ":string ,  "name": string ,  "cost":real ,  "purchaceAmount": int,  "addDate": date ,  "publishDate":date  } |  | GET | newMonthDisks | 5 |
| We use GET because we want to receive information from the server. | Return all the Disks in the system.  Disk Json:  {  "pID":int ,  " artist ":string ,  "name": string ,  "cost":real ,  "purchaceAmount": int,  "addDate": date ,  "publishDate":date  } |  | GET | allDisks | 6 |
| Disks are related to the seller, not to the client.  There is no information to deliver to the server | Returns all the Disks of the category in the system.  Disk Json:  {  "pID":int ,  " artist ":string ,  "name": string ,  "cost":real ,  "purchaceAmount": int,  "addDate": date ,  "publishDate":date  } | GET parameters:  “category”= category | GET | disksByCategory | 7 |
| Disks are related to the seller, not to the client.  There is no information to deliver to the server | Returns all the Disks of the catagory in the system sorted by "sortType".  (if catagory equals “all”, return all the Disks in the system sorted by "sortType". )  Disk Json:  {  "pID":int ,  " artist ":string ,  "name": string ,  "cost":real ,  "purchaceAmount": int,  "addDate": date ,  "publishDate":date  } | GET parameters:  “category”= category  “sortType”= sortType | GET | sortedDisks | 8 |
| Using GET because only one parameter is being transferred.  There is no information to deliver to the server | Returns all the Disks that are recommended for the specific user.  Json:  {  "diskList":[array of pID]  "category":string  } | GET parameters:  “username”= username | GET | recommendedDisks | 9 |
| Disks are related to the seller, not to the client.  There is no information to deliver to the server | Returns all the Disks that fits the user's search sorted by publishDate.  Disk Json:  {  "pID":int ,  "artist":string ,  "name": string ,  "cost":real ,  "purchaceAmount": int,  "addDate": date ,  "publishDate":date  } | GET parameters:  “artistName”= artistName | GET | DisksByArtist | 10 |
| Using GET because only one parameter is being transferred.  There is no information to deliver to the server | Returns all the previews orders of the specific user.  DiskInOrder Json:  {  "oID":int,  "orderDate":datetime,  "shipmentDate": date,  "currency": string,  "totalCost": real,  "DiskList[Disk]  {  "quantity":int,  "artist": string,  "name": string,  "cost":real,  "purchaceAmount":int,  "addDate":date ,  "publishDate":date  }  } | GET parameters:  “username”= username | GET | userOrdersHisotry | 11 |
| cannot pass in GET.  There is information to deliver to the server and also update the db. | Return orderID  (If purchase did not done successfully then orderID =-1) | Order json:  {    "username": string,    "currency": string,    "shipmentDate": date,    "totalCost": int,    "disksAndQuantityList": [“pID”:int,”quantity”:int]  } | POST | purchase | 12 |
|  | Returns: bool, amount.  if the Disk quantity that the user want to buy is avilable then stockAmount is 0 because it means nothing to the user/client. Else stockAmount is the maximum amount that the user can buy.  Json:  {  "isInStock": bool,  "stockAmount": int  } | GET parameters:  “pID”= pID  “quantity”= quantity | GET | isInStock | 13 |

הערות:

1. בבקשות בהן הclient מבקש מהserver מידע על דיסקים, אנו לא שולחים לclient את המלאי של הדיסק כיוון שזהו מידע שאמור להיות חסוי למשתמש (משתמש שילחץ על F12 יראה את התוצאה).

במקום זה הclient ישתמש ב: isInStock לפני רכישה של דיסק / לפני הוספה לעגלה.

1. שליפת דיסקים (חיפוש) מתבצעת לפי אמן (ולא לפי שם דיסק) כיוון שיותר נפוץ בשוק לחפש לפי אמן ולא לפי שם דיסק.